CHAPTER 1

The Sound of Inevitability

There have been very few fundamental changes in computing.

On the surface, that may sound like the statement of a madman, or perhaps at least someone from an alternate universe. Nonetheless, it is true.

Sure there have been, are, and will likely continue to be a nearly incomprehensible fire hose of particular changes, some rather flashy in and of themselves. Simple things like pocket-sized flash drives that store more than the corporate mainframes of 30 years ago, or perhaps ubiquitous mobile devices for everything from the mundanely practical—e-mail, calendars, and contacts—to the cheerfully sublime. Much more complex developments such as the open source movement; the advent of relational databases; and the rise (and fall) of whole operating systems and their surrounding ecosystems, even those whose perpetual dominance once seemed assured (how many desktop machines are running CP/M these days?). These have come and gone, perhaps lingering in some niche, forgotten by all but a few fanatical devotees.

But truly fundamental change—the tectonic shift that literally changes our landscape—happens only once in a long while, perhaps every ten or more years, even in the computing business. Fundamental change of this magnitude requires a number of smaller innovations to pile up until a true nexus is reached, and we all start marching down a different road.

Of course, as historians are fond of lecturing the rest of us mere mortals, these sort of fundamental changes are nearly impossible to

recognize while we are in the middle of them, even as they loom imminently.

When researchers at the University of Pennsylvania were feverishly working on ENIAC—generally recognized as the first programmable, general-purpose electronic computer—as the future of the world hung in the balance in the midst of World War II, do you think they envisioned computers embedded in nearly everything, from greeting cards to automobiles, from microwaves to MRIs? When researchers at the University of California, Los Angeles, and elsewhere in the midst of the Cold War strove to make computer networks more resilient in the face of nuclear attack,[1] do you think any of them envisioned the Internet as we see it today? Likewise, when Tim Berners-Lee and other researchers at CERN were trying to come up with an easy way to create and display content over this new, literally nuclear-grade network, do you think they envisioned the impact on everyday life (both personal and professional) their new creation would have, or even the simple breadth and depth of stuff—from the sublime to the silly—that would be available on this new, supercharged ''Internet'' ? One estimate is that there are more than 500 exabytes—that's 500 *billion* gigabytes—in this ''digital universe,'' and that this will double every 18 months.[2]

The simple truth is that very few, if any, of the people involved in these developments had much of an idea of the consequences of their creations, of the impact on our personal lives, our culture, even the society in which we live—from how we interact with our families to how we conduct business.

Whether you are ''technologically modest,'' or are either by age or temperament not ashamed to let it be known, at least in certain circles, that you are a bit of a geek . . . either way, it is pretty much a given that developments in computing are having a big impact on our society, and more to the point, an even bigger impact on how we conduct our business.

And bigger changes—tectonic shift–scale changes—will have at least commensurate impact on our lives in every dimension, including the fields of commerce. One example, perhaps a seemingly simple one, yet central to many of the changes now underway, will suffice to illustrate this point.

Consider for a moment newspapers. We now face the very real prospect—actually the near-certainty—of at least one (and probably many) major metropolitan area in the United States without a

traditional (local, general purpose, print, widely circulated) newspaper. While this eventuality may be stayed—perhaps for quite some time—via government intervention, the fact that this will eventually occur is not in doubt. In a culture still echoing with such reporteresque icons as Clark Kent, or at least the more prosaic Bernstein and Woodward, this was once unthinkable. Now it is simply inevitable.

There was a time when the technology of newspapers—cheap newsprint (paper), high volume printing presses, delivery networks including everything from trucks to kids on bicycles—was the only reasonable means for mass distribution of information. In fact, with help from some of the newer technologies there was even a new national newspaper (*USA Today*) founded in the United States as late as 1982. But with the advent of alternative delivery channels—first radio, then broadcast cable, and satellite television—increasing amounts of pressure were put on the newspapers.

The immediacy of the newer channels led to the widespread death of afternoon newspapers in most markets; anything delivered to the dinner table in a physical paper was hopelessly out of date with the evening news on television or radio. The morning papers had the advantage of broad coverage collected while most people slept, and as a result have held on longer.

However, at the same time intrinsic limitations of the newer technologies made them better for certain types of information, though not as useful for others. For example, a two-minute video from a war zone could convey the brutal reality of combat far more effectively than reams of newsprint, but did little to describe the complex strategic elements—political, economic, cultural—of the conflict itself. As a result, a certain stasis had been reached in which newspapers carved out what appeared to be a sustainable role in the delivery of news.

Then came the Internet.

In particular, the effectively free and ubiquitous—and yes, near-instantaneous—delivery of all sorts of information mortally wounded the newspaper business. As the first round of the web ecosystem grew, the only remaining stronghold of the traditional newspapers—their ad-based revenue model—was made largely irrelevant. eBay, Craigslist, and freecycle (among others) replaced the classifieds, and online ads took out most of what was left.

Some newspapers will undoubtedly manage the transition in some manner or another, perhaps even emerging as something

fairly recognizable—particularly national/international properties such as the *Wall Street Journal* and the previously mentioned *USA Today*—and perhaps even financially sound.

But those that do will likely largely do so without their original distribution technologies, and more important, many will not make the transition at all.

All of this upheaval in news delivery—the enormous changes that have already occurred and that which is yet to come—have been enabled by developments in computing technologies, with the widespread adoption of everything from the Internet to the iPhone. It is probably worth remembering that all of this has occurred largely without cloud computing, and as a result we are probably less than 10% of the way through this transition in news delivery, and this is only one industry. One industry, one example, with entire economies yet to transform.

Even so, some things have not changed much, even in the delivery of news. The computing infrastructures range from the stodgy (server, even mainframe-based systems within many newspapers) to circa-2009 state of the art (which we might as well start referring to as ''legacy web,'' web 2.0, old-school web, something like that). By and large these systems still cost too much to acquire, do not adapt to changes in demand nearly easily enough, are not reliable enough, and remain way too complex and costly to operate. Even the few systems that do not suffer from all of these problems are not ideal, to say the least: Some are proprietary, and most are either too complex to create new application software, or simply do not scale well enough, at least for the sort of software that researchers are hard at work developing. In particular, with the first generation of electronic news infrastructures focused on just *delivering* the news, the next generation will be focused on sifting through all of that content, looking for just the right stuff.

All of that sifting and sorting and searching will take orders of magnitude more computing capacity than we have anywhere today. How will we pay for hundreds and thousands, perhaps even tens of thousands *times* more servers and storage than we have today—almost unimaginable quantities of computing? How will we operate them? Write new software for them? It is fair to wonder how we will even power all that gear. Assuming that all of these concerns are resolved, then, we will face a larger question still, one which we

presume has many answers: What sort of business models are enabled by all of this, and how do we get there?

Before we leave this example, it is probably worth considering our present circumstances just a bit more. In particular, most of the history of both economics and engineering can be understood by thinking about managing *scarcity.* In other words, how do I get the most done with the least stuff, or within certain limits? For example, that underlying drive to dealing with scarcity, at its core, drives the startup team to work harder and pay less, the Fortune 500 enterprise to optimize manufacturing processes, and entire nations to set energy policies. Allocating scarcity is just Economics 101. Of course, it is also Engineering 101. Dealing with scarcity causes communications engineers to develop better video compression schemes, improve CPU designs to get more done in the same amount of time, and even rethink server packaging to reduce power consumption and labor costs.

While scarcity may be the nemesis of some, it is quite literally a prime mover behind the developments that have together come to be known as cloud computing. What does this mean, and how can it be possible?

A Persistent Vision

Better, faster, cheaper is often heard in technology circles. More than a policy, more than a philosophy, this is literally a way of life within technology communities. In an ideal world imagine that:

Computing—computation, storage, communication—is relatively free, scales up or down as needed, scales as much as needed, operates itself, and always works.

To one degree or another, this is the persistent vision that drives many of those who are developing cloud computing. Is all of this presently possible? Of course not; yet we are inexorably on this path.

Achieving this vision is, of course, a complex endeavor with far more to it than may meet the eye at first glance. That is why there is the rest of this book, for starters!

Before we go further let us elaborate a bit on the dimensions of this vision.

Engineers and mathematicians talk about something being ‘‘within epsilon of zero.’’ This is a term that comes from calculus. It

simply means the process of approaching a particular limit, from wherever you started to the limit itself. In the case of the cost of computing infrastructure, that limit is zero. For most of computing history the costs of infrastructure have dominated decisions about what to deploy when: How much will those servers cost? How about that storage farm? That network? Now, however, we can start thinking about those costs being ''within epsilon of zero''; that is, over time the computing infrastructure comes closer and closer to being free. That leaves other costs as the new, more significant considerations—software licensing, data acquisition, for just two examples—and this will be examined more closely later in the book.

A Little History

In one sense the evolution of computing has been one long blur, with change piling on change, products that are ''long in the tooth'' in less than a year and virtually classic soon after, and with new concepts—Moore's Law, for example—created simply so that we can describe, understand, and effectively institutionalize this relentless rate of change.

But there are times when these changes pile up in such number, in particular combinations of new capabilities and logical consequences, that the whole industry does head off in a new direction—when the very conversations, the underlying concepts, even the possibilities themselves change.

To help understand the import of our current transition into a computing world dominated by cloud computing, think a bit about where we have been, where we are now (at least just slightly before exactly right now), and both how and why we have travelled these paths. While there are clearly many ways that the history of computing can be written, this one will only focus on the big changes—the nexi[3] themselves—where the very possibilities change.

Three Ages of Computing

While there many ways to get a handle on the evolution of computing, in order to gain an initial understanding just where cloud computing fits, of just how significant and, yes, disruptive it is and will be, it is sufficient to consider the broad sweep of computing history.

First Age

Think about the role of computing within the typical organization prior to the widespread adoption of the Internet. The focus was on automating particular operations, creating supporting business processes, and of course, always improving efficiency.

Notice that the focus was *within* individual organizations, by and large. Yes there were purpose-built networks for interacting between organizations, some of them even fairly large and important (stock trading and manufacturer-specific EDI [electronic data interchange] networks are two notable examples), and even for certain organizations to interact with their customers (e.g., credit card authorization networks), but each of these tended to have a very specific, rather narrow focus. Even more important, these examples were relatively few and far between, and very difficult to achieve.

This was the first age of computing, in which organizations looked internally for the big wins. For the most part the edges of each organization remained the same as they had always been.

At the beginning of the first age the focus was on big infrastructure—mainframes, big point-to-point networks, centralized databases, and big batch jobs. Toward the end, terminals evolved into personal computers, networks went from hierarchical (with the mainframes at the center of each network) to decentralized, with a broader, generally more numerous collection of servers and storage scattered throughout an organization. While batch work still existed, many programs became interactive through this first age, eventually gaining much more visual interfaces along the way.

Infrastructure tended to be associated with particular applications—a practice since pejoratively known as ''application silos''—and important applications generally demanded enterprise-grade (read: expensive) infrastructure—mainframes or big servers, and so forth.

Application architectures tended to follow the same evolutionary path, with earlier applications being generally centralized, large and heavy, while client-server and distributed application architectures became mainstream toward the end.

This period also saw the rise of databases, along with the beginnings of specialized storage infrastructure upon which those databases relied.

Technologies such as parallel computing, artificial intelligence, and even semantic processing remained exotic tools that were employed in only the most demanding problems, where ''cost was no object'' (at least in theory), where the goal was simply to solve ever-bigger, ever-thornier problems—places like the nuclear weapons laboratories, national intelligence agencies, scientific research institutions, and the like.

Despite the rapid, consistent improvements in individual hardware and software technologies throughout this period, the limitations and complaints remained nearly constant. In particular, no matter how much was poured into the IT budget, the foul nemesis of ''application backlog'' was heard in the hallways of nearly every enterprise. Who did not constantly complain about how much IT was costing?

Still, it was at least (generally speaking) possible to automate crucial operations within a company, and as a result overall corporate efficiency steadily increased. More autos were made with less labor, more packages delivered with the same number of employees, higher revenues per store per employee, and so forth.

This period covered about four decades, from the roots of enterprise computing in the 1950s until the rise of the Internet in the mid-1990s. As with all major shifts in a society, its culture and technology, the roots of the end of the first age of computing were sown years before the second age began.

Second Age

The second age of computing is really the story of the rise of the Internet—Sun, Cisco, Mosaic (which became Netscape), web 1.0, eBay, Yahoo, baby.com, and the first Internet Bubble—all of it, good and bad, all of the tumultuous commotion of the first Internet land rush.

While many advances contributed to the beginning of the second age, the two most crucial were the development of the Internet itself, and the development and near-ubiquity of easy-to-use, visually attractive devices that could be used by nearly everyone.

The story of the development of the Internet is well known[4]—starting from a research question (Can we build a more resilient network, one that can survive a nuclear attack?), to a more loosely coupled set of higher level communications protocols (e.g., ftp for

file transfers, smtp for e-mail, http for web content) built on top of this newly resilient foundation, then to a whole ecosystem of new software. From browsers to web servers, among many others, the Internet quickly went from ''who cares?'' to ''must have!''. By the early 1990s this new, sort of crazy idea began to dominate even mainstream business thought, to the point that normally sane, rational people predicted such improbably outcomes as the elimination of all brick-and-mortar stores, the irrelevance of a nation's manufacturing base, and in some cases the irrelevance of nations themselves.

This in turn led to truly historic business hysteria: the Internet Bubble. (Truth be told, if not for macro-level economic problems that started in late 2008 the onset of cloud computing may have triggered Internet Bubble 2.0.)

But as the dust settled and all calmed down, it was clear that the world had shifted. Any enterprise intending to prosper now had to consider how best to reach their customers and their ecosystem of suppliers, and where to look for their newest competitors, all in the face of the newest reality—ubiquitous connectivity.

Likewise, the ubiquity of visually rich devices—at first stationary, then evolving to include the ''handheld slabs of glass'' (iPhone, android phones, Palm pre, and their successors) made it possible for the non-geek to care. While command lines and text terminals were enough for many of the early adopters, the simple reality is that audience is, by definition, limited.

There were people—including one of the authors—who went from cards, to command line, to modern bit-mapped displays (along with a mouse, laser printer, and local area network, all part of the experimental Alto workstations from Xerox PARC[5]), all well within the span of a single year—1979. At the beginning of that year most work was done on a mainframe via cards, printers, and batch jobs; halfway through 1979 work moved to interactive command-line access via dumb terminals; and by the end of the year you could sit in front of a Xerox Altos, mesmerized by mice, bit-mapped displays, and early networked games (Mazewars[6] being a great example).

While both of these trace their earliest roots—at least in forms that we would largely recognize today—to the mid-1970s, they each took 15 to 20 years to gestate sufficiently to have broad impact.

Overall, the biggest technical contribution of the second age was perhaps the network itself. Forced to deal with the possibility of massive network failures caused by a nuclear attack, researchers

endowed their invention with the ability to self-organize, to seek out alternate routes for traffic, to adapt to all sorts of unforeseen circumstances.

In doing so (perhaps with only partial intent) these researchers removed the single point of failure that was typical of mainframe-inspired networks: and as a consequence in one fell swoop they removed the biggest technological barrier to scaling–the mainframe-centric network itself. Even more telling, foreshadowing changes that would usher in the third age-when they enabled the networks to take care of themselves-these researchers also removed the biggest obstacle to growth—they made these new networks *much* easier to operate.

It is hard to overestimate the importance of two fundamental realities: (1) with the Internet it was now true that everyone was connected to everyone else, anytime, anywhere; and (2) with the ubiquity of visually attractive devices, the data and services available over that pervasive network could actually be used by mere mortals.

Typical technologies included the J2EE application servers (often in clusters) along with relational databases, themselves often in clusters. Developers and researchers everywhere strove to stretch, push, pull, morph—everything but blowing them up and starting over—to make these application architectures more flexible, scalable, more resilient to failure, and so forth, but were mostly unsuccessful, or at least not successful enough.

There were plenty of innovations in software architectures, ranging from improved data techniques to the first forays into what became service-oriented architectures in the early part of the new millennia.

But what had not changed? Far too much remained as it always had, as things turned out. For starters, infrastructure remained expensive, chunky, siloed, and by modern standards phenomenally overengineered (after all, the infrastructure really should not fail), and consequently even more expensive. Great strides were being made in distributed software architectures, but (outside of the foundational TCP/IP networks themselves) most applications and infrastructure software remained difficult to configure, complex to create, and brittle when faced with failure. As a result, operations remained enormously difficult and therefore both costly and error prone, which in the final analysis was the cruelest constant reality of all.

Before we continue in this narrative, let us take a step back to consider two more constants in computing—the drive for ever-increasing scale and the drive for ever-lower expenditures (i.e., the ''drive for cheap'').

Drive for Scale Remember back to the middle of the first age, in the 1970s and 1980s—most computing was done on relatively mundane, large-scale individual computers, or perhaps in small clusters of relatively big machines. Even then, for the researchers, scientists, or perhaps intelligence agencies who were simply trying to solve the biggest problems possible, this was never enough; for that matter, nothing was ever enough, no matter how big and fast. Those folks were the ones who were exploring the edges of parallel computing and distributed architectures, who were thinking of highly pipelined supercomputers and vector processors.

Yet in the mid-1980s another thread of investigation took root—inspired by biological systems themselves—which started by combining large numbers of relatively slow computers, sometimes loosely coupled via a local area network (these came to be often known as grids) and sometimes linked internally via specialized connections (such as the exotic Connection Machine 1, produced by Thinking Machines, Inc., which was the effort to commercialize the doctoral work of Daniel Hillis). In all cases these alternative architectures were difficult to develop software for, cranky to operate, and enormously expensive. Even though most of those efforts eventually evaporated, they did at least make one very important contribution: They showed that it was indeed possible, particularly for certain applications, to build very large computing facilities out of very modest components.

This drive for scale went mainstream along with the Internet. This was true in many dimensions, but for one easy example just think of the indexing problem itself—whereas an early (circa 1994) Yahoo index might have had less than a hundred, or at most a few hundred entries, and could be manually created, by the beginning of 1995 the number of web sites was doubling every 53 days[7] and was passing anyone's ability to manually index. This growth then created the need for computing infrastructures that could scale at the same rates or faster, as well as application and data storage architectures that could also scale apace.

Yet there was one fly in the ointment that occurred about this same time—the silicon companies (Intel, competitors, and friends)

began to reach their practical limit for scaling individual execution units (which came to be known as ''cores''). In fact, this problem had been looming for some time, but the processor designers tended to solve the problem the way they had always done: Throw more hardware at it and hope it would go away. In late 2004 Intel announced that they were largely abandoning their push to increase the ''clock speed'' of individual processing elements, and going forward would instead be, increasing the number of individual processing units (or cores). While, at least in theory, this drive for increased core counts can deliver the same raw computing capacity, in practice it is much more difficult to write application software that can make use of all of these cores.

This is, in essence, the ''parallelization problem,'' which in many ways is the same no matter whether you are writing software for multiple cores within a single piece of silicon, multiple cores on multiple processors within a single computing system, or multiple cores on multiple processors on multiple computing systems within a single grid/cluster/fabric/cloud.

Sound complex? To be honest, it is—successfully writing a parallelizable application can be enormously complex, difficult to do well, even more difficult to do reliably, and more difficult still to make it also easy to operate. In other words, the silicon and systems designers had punted, shifting the burden for scaling to the application software and operational communities.

Drive for Cheap Of course one drive that remains true in every age and in every domain is the drive to reduce costs—cost to acquire, cost to deploy, cost to operate, cost here, cost there, cost anywhere—just reduce them all.

In the midst of the rubble of the first Internet Bubble (bursting), many different groups began to wonder just how to make use of these increasingly capable commodity computers for problems that we really cared about—mission-critical problems, the ones that ''absolutely, positively, have to work.''[8]

For example, the roots of Appistry (a company founded by one of the authors) lie in just such a question. When building a digital recording studio out of purely commodity parts (no label, cheapest fastest stuff that money could buy), after running benchmarks the obvious question came up: Why are we not using cheap stuff like

this (meaning the plain label, pure commodity computing parts) for problems that ''we really care about''?

The answers to that question—how to ensure that commodity infrastructure could be ultimately reliable, easy to operate, easy to bring software into and so on—led to multiple patents, products, and companies, and is a question whose answers are definitely worthwhile.

The economics of utilizing commodity components are compelling, if—and only if—you can safely answer those key questions. The economies of scale with commodity infrastructure, such as general-purpose processors, are simply overwhelming when compared to specialty designs. It is common for a collection of commodity computers to deliver the same capacity for less than 10% of the cost—sometimes far less than 10%—of enterprise-grade servers and mainframes.

It is no longer a question of ''is this possible,'' but rather ''how, when, and where.''

That same question—How can we use commodity infrastructure for problems that we care about?—is being asked and answered in various ways by forward-thinking technologists and executives everywhere in the relentless pursuit for ''cheaper, faster, better,'' and is integral in the transitions to cloud.

Third Age

Now let us resume our narrative. Early in the second age Yahoo had made a name for itself by ''indexing the Internet,'' which for some time was mostly manually done. While this was sufficient for a while, it soon became apparent that manually built indices could never keep up with the growth of the Internet itself.

Several other indexing efforts began, including AltaVista, Google, and others, but it was Google that brought everything together. While a full understanding of why Google became so dominant–at least as of this writing-is beyond the scope of this book, several key factors can be easily understood.

- First, the collection of data about the current state of the Internet, and the processing of that data had to be as absolutely automated as possible.

- In order to save as much money as possible, the infrastructure would be constructed out of commodity components, out of ''cheap stuff that breaks.''
- Data storage needed to be done in a simple, yet fairly reliable manner to facilitate scaling (the Google File System, or GFS—notice the lack of a traditional database, but more on that later).
- New types of application development architecture(s) would be required, which came to include the so-called map-reduce family (which inspired open source descendants such as Hadoop) among others.
- Operations needed to be as automatic and dependable as possible.
- Outages in the application were tolerable; after all this was search, and who would miss a few results if an outage occurred?

So almost before anyone really knew what was happening, in order to scale a basic search facility and do so cheaply, Google had created much of what we could probably first recognize as a cloud.

Another interesting case is Amazon. In the first six or seven years Amazon largely built its computing infrastructure the traditional way, out of big, heavy servers, with traditional relational databases scattered liberally throughout. That was fine in the early days, and definitely fine during the first couple of years after the Internet Bubble burst (particularly since much high-end hardware could be had for pennies on the dollar after the first bubble), but as commerce on the Internet began to gain some real momentum it became abundantly clear that the Amazon computing architecture(s) had to change.

At the same time, in order to build customer and vendor stickiness Amazon had begun exposing individual services, even select customer data as callable services—one of the key application lessons that is leading to the third age—and so had accelerated decomposing many of their applications into dozens, or sometimes hundreds, of individually callable services.

About that time (2001–2003) Amazon began to adopt many of the same principles as Google had done early on, but then they took things a step further. Instead of simply offering entire services such as search, e-mail, maps, photo, and so forth with various

services exposed for calling from outside, in 2006 Amazon began to offer basic computing resources: computing, storage, and network bandwidth in highly flexible, easily provisioned, services, all of which could be paid for ''by the drink.''

Others offered public cloud services that made certain unique contributions, including Salesforce.com, which was probably the first public cloud service that was targeted at the enterprise customer and required those customers to store very sensitive data outside of their own facilities. While many thought that sale was not doable, that no enterprise large or small would risk their customer data on something so unproven, the allure of an easy, pay as you go CRM (customer relationship management) implementation led to the rise of Salesforce.com (and competitors, in the sincerest form of flattery), emphatically proving otherwise, that the enterprise customer could trust these services. That their initial rise to meaningful market share and then eventual dominance came largely at the expense of the traditional, install-in-your-own-shop application with an overwrought, often painful, and unintentionally costly implementation was simply a bonus.

While each of these examples have their roots firmly in the middle of the second age, either their original or subsequent decisions played crucial roles in bringing together the beginning of the third age, the age of cloud computing.

It is during this era that that persistent vision that we discussed earlier can finally begin to become true:

Computing—computation, storage, communication—is relatively free, scales up or down as needed, scales as much as needed, operates itself, and always works.

With that in mind, let us step back and take a look at some of the particular developments that are enabling this persistent vision to begin to become reality.

Broad Enablers

Over the course of the 1980s and 1990s there were key advances that came together to enable the transition to the cloud computing era—the third age. We are at the cusp of this transition as we complete the first decade of the new millennium. While not a comprehensive list, these are some of the more notable enablers:

- **Commodity Hardware.** In the three basic areas of computing components—chips (processors, memory, etc.), storage (mostly disc drives), and network (both within a datacenter, wide area, and wireless)—there have been large strides made in the capabilities of what is by historical standards throw-away equipment. For example, a client of one of the authors was able to match a competitor's industry-leading, mainframe-based performance in processing high-volume customer transaction with less than a dozen cheap commodity boxes sitting on a repurposed kitchen rack. Total bill? Less than $10,000. Yes it works, and it works very well. The key, of course, was in how the applications were constructed and how that set of machines is reliably managed. In any case, there will be more on this example as well as others later in the book.
- **Network Speed.** While network performance has not increased at the same rate as either processor or storage performance (which will lead to interesting problems as clouds develop—we will cover the details of this in depth in Chapter 8, All Things Data), huge strides have been made in both the connections within a datacenter and those outside.

  For example, by the time you are reading this a ''gigE'' network card (for use by a commodity computer within a datacenter) will be less than $10 each in small quantities. To put that in perspective, that is *about 400% faster than the internal bus connections*[9] (the key internal connectivity within server computers) of the typical big servers of the early 1980s. Also as you are reading this, a 10 Mbps wired connection for the home or office will average less than $50 per month in the United States, and even less than that in many parts of the world. Mainstream mobile wireless (for those ubiquitous ''slab of glass'' mobile devices that make accessing all these services so pleasant) speeds will be closer to 7 Mbps, at the cost of only a modest part of the typical monthly cell phone budget. The point is simple: Whether within the datacenter, at fixed locations throughout the world, or on mobile devices, cheap, fast, reliable, and ubiquitous network connections are a fact of life.
- **Virtualization.** Virtualization started as a way to share the use of very expensive mainframes among otherwise incompatible operating systems, then flowered in the later similar trend to consolidate large numbers of small servers (each typically

dedicated to one or two specific applications). It is the ability to operate particular resources (such as computers, networks, and so forth) largely independent of the physical infrastructure upon which they are deployed. This can be a tremendous boon for operations.

For example, the initial configuration of the operating system for a server, along with the applications to run on that server can take hours, if not days. With virtualization that initial work is done once and the results put on the shelf, to be deployed onto physical hardware when needed. This process, sometimes referred to as *hydration*, can be done in as little as a few seconds to minutes and repeated as often as needed, thereby enabling the possibility of easily deploying basic software to large numbers of computers.

- **Application Architectures.** Beginning with the development of *object-oriented* languages and tools in the 1980s and 1990s, and continuing on through the beginning of web services and service-oriented architectures during this decade, software architectures have made many strides toward the eternal goal of software reusability, itself driven by the desire to make it easier to construct software. A key characteristic of typical cloud applications has been the fine-grained components, with an exposed application programming interface (API) or interface (i.e., the ability to make use of that portion of an application from nearly anywhere on the Internet—any place that makes sense, and probably even a few places that are just for show!).

  This ability to mix and match relatively independent software services is crucial in making software more useful. For many, this has been the practical realization of service-oriented architectures (SOA), an interesting topic that we will explore in more detail later the book. (A more detailed discussion of the relationship between SOA and Cloud, and industry adoption trends for both, is explored in detail in Chapter 5.)

  In addition, there have been significant advances in creating more resilient, self-organizing application platforms that are inherently at home on top of the very fluid, commoditized infrastructure typical in cloud computing.

  Finally, the need to become more adept at parallelization in order to effectively use multi-core processors is beginning to have an impact.

- **Data Storage Architectures.** The first two ages of computing were very much dominated (for very good reasons) by the database systems—relational databases such as Oracle, MySQL, SQLServer, Postgress, and others. Entire (data management) organizations exist in most enterprises to manage the structure of this relational data within these repositories; with strict rules about how such data is accessed, updated, and so forth. Unfortunately, what we have learned from abundant experience is that at some point the block to scaling any given application will nearly always be the relational database itself.

  As a result the whole approach to reliably storing, processing, and managing data at large scale is being rethought, resulting in a number of innovative, novel technologies that show significant promise.

  We are also beginning to see some significant improvements in the underlying storage infrastructure itself, both in composition and operations. In any case, this whole area will be explored in much more depth in Chapter 8, All Things Data.
- **Pervasive High Quality Access.** The reality—quality, variety, quantity—of high quality, visually attractive, widely available devices has had a tremendous impact on the development of cloud computing. Typical devices include fixed desktops with one or more flat panels; laptops and netbooks of every size, price range, and performance; ubiquitous, sometimes specialized, and nearly always relatively inexpensive handheld devices such as the iPhone and its growing range of competitors (such as the rapidly-expanding range of devices running the Android operating system from Google). Perhaps most importantly, these devices share in common a wide range of wireless high-speed Internet access.

  Taking all this into account, this plethora of high quality, pervasive, always-connected devices has greatly increased the number of customers for services and content—the data and applications sold on the cloud—and has also increased each customer's appetite for even more services and data.

  Consider one small example. In March 2008 Apple announced that they would create a marketplace from which third-party developers could sell applications to owners of an iPhone. Despite a tremendous amount of uncertainty—

including many who thought that the whole concept would simply fizzle out for any of a number of reasons—within the first nine months Apple was able to sell[10] more than one billion individual applications; the second billion came in about six months; the third in just over three months. From zero to a billion in less than a year, then six months, then three, then . . . well, regardless of what is next, that is a nearly incomprehensible rate of growth, a reality worth pondering for a moment.

- **Culture.** We have become conditioned by the expectation (quite often the reality as well) that everything is available all the time—that Google and others will be able to tell you where any place is, and that you can then reach your friends (no matter the time or place) to tell them about it. Perhaps all that seems too obvious to think about much anymore, but take a moment and ponder what this assumption, this daily reality, has wrought on society. While an in-depth study of this phenomenon is outside the scope of this book, it is such an important factor that it must be considered.

  After all, in some sense culture is a measure of how members of a society interact with each other, and the transition to the era of cloud computing is bringing incalculable changes to this very arena.

  That in our culture—for that matter, nearly any culture around the world today—this means of communication is simply a given. The fact that we all take it for granted is a profound enabler for future services, for future proliferation of cloud-based services.

  For example, consider that even such a venerable, ancient institution as the Catholic Church has launched a number of initiatives in the social media (including Facebook, Twitter, YouTube, and more), and the present Pope has even encouraged young people to ''evangelize the Gospel into these new societies'' and ''into this new continent'' (speaking of the communities that are formed in the various social networks), and that this is ''a very high priority.''

  Ponder this carefully: If a 2,000-year-old institution that has rarely been accused of haste can understand the fundamental nature of these changes and act on them, can any organization afford to do less?

After all, this is what our cultures now expect; this is what people demand; this is how people interact.

To recap, there have been several key factors that have enabled the development of cloud computing at this time, in this place. Now let us turn our attention to what some of these early clouds have contributed to our understanding of cloud computing, of just what is possible.

Big Contributions

While most of these enablers came about for different reasons, it has really been the combination of ''all of the above'' that enabled cloud computing to get started. Once started, of course, the pace of innovation began to increase significantly, and that increase in the rate of change is itself continuing to increase—that whole ''critical mass'' notion all over again.

For example, once actual clouds began to be deployed, utilized, liked, and then as a result scaled up even more, then the early ideas about cloud-optimal application architectures needed to be advanced even further. Likewise, the very need to scale has pushed the innovative data models even further, which enables more scale.

Yet in addition to this self-fulfilling innovation there have been a few, perhaps unexpected ''bonus'' advances. In particular:

- **Operational Models.** In order to deal with the scale there has been some significant development of novel operational models, with a degree of automation far beyond any prior.
- **Flexibility.** From the beginning these infrastructures needed to be able to scale up easily; then it became clear that they also need to be able to scale down just as easily. This led to automated mechanisms for requesting more and releasing unused infrastructure, and in some cases going further and simply allowing policy-driven, automated scale changes with no human intervention. These capabilities are wrapped in APIs and available via the web, of course.
- **Consistent Infrastructure.** In order to facilitate the scale, flexibility, and ease of operations most clouds have a relatively small number of physical building blocks from which they are constructed. Rather than the hundreds of unique servers,

versions of servers, and configurations of versions of servers that populate the typical, pre-cloud datacenter, even the largest cloud computing datacenter may have no more than a handful, perhaps as few as three or four different possibilities.

- **Packaging and Construction.** With consistency a given, the next step is to consider more efficient, higher density packaging and datacenter construction. Innovations in this area include everything from the very cheap—naked computer motherboards mounted open air (no cases) on sheets of plywood, all arranged in stacks and rows—to the more costly (though highly engineered for space and cooling efficiency)—stackable, semi-trailer sized containers stuffed full of the actual computing infrastructure. There are now entire datacenters designed to accept a number of these preconfigured, stackable containers, with hardly a month or two passing before someone presents yet a more efficient, even more radical, highly scalable modular datacenter.

All of these advances work with each other, in turn depending on another and then enabling yet another. The interactions are fascinating and useful, and will be explored in more detail in Chapter 2, Concepts, Terminology, and Standards, and again in Chapter 8, All Things Data, and Chapter 9, Why Inevitability Is . . . Inevitable.

Limitations

Of course, with all of the excitement there remain substantive limitations. In particular, much of the early thought leadership (such as in *The Big Switch,* a seminal cloud computing book by Nicholas Carr[11]), or in the ideas contained in ''Redshift Computing,'' a set of concepts put forth by Greg Papadopolous, then Chief Technology Officer of Sun (prior to the acquisition of Sun by Oracle), who claimed that all computing would eventually go into a small number of extremely large public clouds. Papadopolous went so far as to initially predict that eventually there will only be, in essence, five computers!

Upon calming down a bit and thinking through the implications a little more clearly, it began to become clear that while public clouds of various types will play very important roles as the cloud computing landscape develops, they will not work alone. Rather, the public clouds will interoperate and work interchangeably

(where appropriate) with private clouds built and operated on the same cloud computing principles. Some organizations will create their own clouds for any of a number of reasons including control, privacy, security, and reliability, among others, or perhaps do so for data issues—data retention, reliability, and access to computing resources (in order to enable even larger scales, better efficiencies, etc.).

The realization of these concerns with the early Utopian vision of a small number of purely public clouds—and nothing else—is leading to the development of a much richer, more textured cloud computing landscape, with variations that can address these and other considerations as necessary, sharing common foundations yet differing where necessary.

The best part? Each organization has a wide range of choices from which to choose, with the ability to pick and choose as each need dictates.

I Want One of Those

As a result of all this—the promise of reduced costs, easier scale, greater flexibility, reduced deployment cycles and more, much more—over the past couple of years it has become very common, almost a litany, across many organizations to say in one form or another, that ''We want what Google and Amazon have, except that we want it inside our organization, while at the same time interoperating and in other ways working very closely with those very public clouds, and we want to be able to use any of these clouds when WE choose, as best suits OUR needs.''

Back to the Future?

A few years ago a friend of ours was given a tour of a large, technology-dependent Fortune 500 company that was trying to win her business. She was taken to what was an extremely large (for those days) datacenter, with row after row of large ''excelsior class'' mainframes (a figure of speech for very large and costly stuff, but it works, particularly here).

The salesperson, obviously proud of the datacenter pointed to all of this ''big iron'' and went on and on about how they could meet her needs, no matter what. They even were double-especially proud of the empty floor space in the datacenter, with tape outlines stretching as far as the eye could see marking the places where

future, planned excelsior-class mainframes would be delivered early and often to handle growth.

''See, we can handle any need you might have.''

Our friend was, on the inside, simply appalled. Turns out that she already had a taste of what cloud computing could be, so all she could see as she looked out across that floor was an enormous amount of fixed costs, with high operational costs at least partially driven by the legions of highly skilled technicians hovering about each precious mainframe, all these costs growing inexorably over time with no end in sight, and very little ability to actually scale up fast if her business did anything like what she was hoping.

Sensing her distraction, the salesperson reiterated that ''they could handle growth with this gargantuan facility, that this Fortune 500 organization was most definitely futureproof.''

No, our still-polite friend thought, I am just not looking at the future . . . this is a monument to the past.

It is just a matter of time.

Notes

1. Leiner, Cerf, et al., A Brief History of the Internet, last revised December 10, 2003; Internet Society: www.isoc.org/internet/history/brief.shtml
2. IDC Digital Universe White Paper, sponsored by EMC, May 2009.
3. Nexi is a slightly stylized plural of nexus—those crucial points where everything converges and fundamental change occurs.
4. A good starting place to find more is the ''Histories of the Internet'' section of the Internet Society site (www.isoc.org), where you will find several excellent histories.
5. Xerox Palo Alto Research Center, research lab made famous by luminaries such as Alan Kay and John Warnock (Adobe) as the home of such innovations as the bit-mapped display, highly visual user interfaces for normal computing, mice, local area networks, WISIWYG editing, Smalltalk, and more—all routinely used everyday now.
6. See A 30 Year Mazewar Retrospective at www.digibarn.com/collections/games/xerox-maze-war/index.html.
7. Kill the 53 Day Meme, Jakob Nielsen's Alertbox for September, 1995.
8. Inspired by and with apologies to that famous FedEx tagline ''When it absolutely, positively, has to be there . . . ''
9. A Vax 11/780 had 1200 nanosecond memory, with a synchronous 32-bit bus.
10. Note that this number includes free as well as paid applications, some of which are either ad-supported or involve generating revenue through some other means.
11. Norton, 2008.